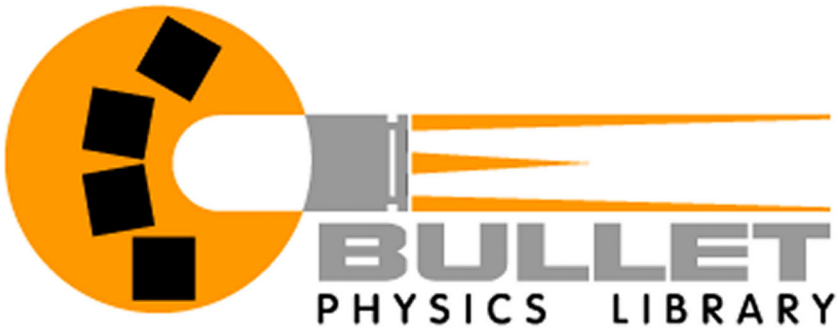


Bullet 2.1 Physics User Manual

last updated by Erwin Coumans on Wednesday, 27 September 2006

Introduction 2
Main Features 2
Download and supporting physics Forum 2
Quickstart 3
- Step 1: Download 3
- Step 2: Building 3
- Step 3: Testing demos 3
- Step 4: Integrating Bullet physics in your application 3
- Step 5 : Integrate only Collision Detection Library without dynamics 3
- Step 6 : Use Snippets like the GJK Closest Point calculation 3

Integration overview 4
- Debugging 4

Basic Demos 5
- CCD Physics Demo 5
- COLLADA Physics Viewer Demo 5
- BSP Demo 6
- Vehicle Demo 6
- Fork Lift Demo 6

Low Level Technical Demos 7
- Collision Interfacing Demo 7
- Collision Demo 7
- User Collision Algorithm 7
- Gjk Convex Cast / Sweep Demo 7
- Continuous Convex Collision 8
- Raytracer Demo 8
- Concave Demo 8
- Simplex Demo 8

Bullet Collision Detection and Physics Architecture 9
- Bullet Collision Detection 10

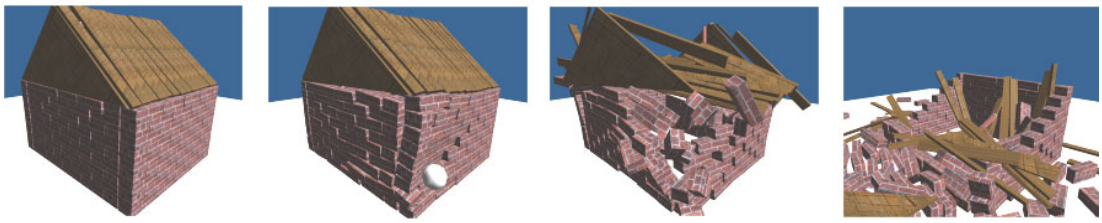
Further documentation and references 11
- Links 11
- Books 11

Introduction

Bullet Physics is an open source collision detection and physics library, related tools, demos, applications and a forum. Supported platforms are Windows (32/64 bits), Mac OS X, Linux, Playstation 3 and others. It is created as toy project by Erwin Coumans, ex-Havok employee and involved in the port of Ageia to Playstation 3. It is free for commercial use under the ZLib license. It is under active development and one of the recent developments is the addition of a GPU Physics and plans for an C# XNA port.

Target audience for this work are game developers, physics enthusiasts and 3d professionals who want to play with collision detection and rigidbody dynamics. There is also a sample integration with COLLADA Physics import and export. 3D Modelers like Maya, Max, XSI, Blender and Ageia's CreateDynamics tools support COLLADA physics xml .dae files. See the References for links.

Bullet is also integrated in the free Blender 3D modeler, http://www.blender.org. The integration allows real-time simulation and also baking the simulation into keyframes for rendering.



Main Features

- ✓ Discrete and Continuous collision detection including ray casting
- ✓ Collision shapes include concave and convex meshes and all basic primitives
- ✓ Rigid body dynamics solver with auto deactivation
- ✓ Generic 6 degree of freedom constraint, hinge and other constraints
- ✓ COLLADA physics import/export with tool chain
- ✓ Open source C++ code under Zlib license and free for commercial use

Download and supporting physics Forum

Primary location to download Bullet and for the supporting physics forum is http://www.continuousphysics.com/Bullet or shorter http://bullet.sf.net

Quickstart

Step 1: Download

Windows developers should download the zipped sources from of Bullet from http://bullet.sf.net. Mac OS X, Linux and other developers should download the gzipped tar archive.

Step 2: Building

Bullet comes with autogenerated Project Files for Microsoft Visual Studio 6, 7, 7.1 and 8. The main Workspace/Solution is located in Bullet/msvc/8/wksbullet.sln (replace 8 with your version).

Under Mac OS X, Linux and other platforms you can compile Bullet using either CMake or jam:

CMake: Download Cmake from http://www.cmake.org and auto-generate projectfiles for Mac OS X Xcode, Linux KDevelop and other build systems.

Jam: Bullet includes jam-2.5 sources from http://www.perforce.com/jam/jam.html. Install jam and run ./configure and then run jam, in the Bullet root directory.

Step 3: Testing demos

Try to run and experiment with CcdPhysicsDemo executable as a starting point. Bullet can be used in several ways, as Full Rigid Body simulation, as Collision Detector Library or Low Level / Snippets like the GJK Closest Point calculation. The Dependencies can be seen in the doxygen documentation under 'Directories'.

Step 4: Integrating Bullet physics in your application

Check out CcdPhysicsDemo how to create a **btDynamicsWorld** , **btCollisionShape** and **btRigidBody**, Stepping the simulation and synchronizing the transform for your graphics object. NOTE: CcdPhysicsEnvironment is obsolete.

Step 5 : Integrate only Collision Detection Library without dynamics

Bullet Collision Detection can also be used without the Dynamics/Extras. Check out the low level demo Collision Interface Demo, in particular the class CollisionWorld. Also in Extras/test_BulletOde.cpp there is a sample Collision Detection integration with Open Dynamics Engine, ODE, http://www.ode.org

Step 6 : Use Snippets like the GJK Closest Point calculation.

Bullet has been designed in a modular way keeping dependencies to a minimum. The ConvexHullDistance demo demonstrates direct use of **GjkPairDetector**.

Integration overview

If you want to use Bullet in your own 3D application, it is best to follow the steps in the CcdPhysicsDemo. In a nutshell:

- ✓ Create a btDiscreteDynamicsWorld object

This btDynamicsWorld is a high level interface that manages your physics objects and constraints. It also implements the update of all objects each frame, including the interaction between collision detection and physics.

- ✓ Create a btRigidBody or btCollisionObject and add it to the btDynamicsWorld

To construct a btRigidBody or btCollisionObject, you need to provide:
- CollisionShape, like a Box, Sphere, Cone, Convex Hull or Triangle Mesh
- Mass and Material properties like friction and restitution
- World transform, btTransform, that controls the graphics representation

- ✓ Update the simulation each frame

Call the stepSimulation on the btDynamicsWorld. Preferably use a fixed timestep of around 60 hertz (1.0/60.0). This will perform the collision detection and physics simulation and updates the world transform.

There is performance functionality like auto deactivation for objects which motion is below a certain treshold.

A lot of the details are demonstrated in the Demos. If you can't find certain functionality, please use the FAQ or the physics Forum on the Bullet website.

Debugging

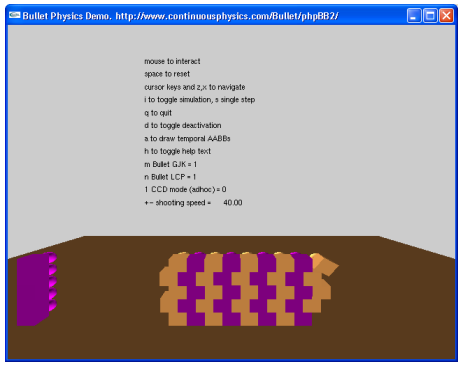
You can get additional debugging feedback by registering a derived class from IDebugDrawer. You just need to hook up 3d line drawing with your graphics renderer. See the CcdPhysicsDemo OpenGLDebugDrawer for an example implementation. It can visualize collision shapes, contact points and more. This can help to find problems in the setup. Also the Raytracer demo shows how to visualize a complex collision shape.

Basic Demos

Bullet includes several demos. They are tested on several platforms and use OpenGL graphics and glut. Some shared functionality like mouse picking and text rendering is provided in the Demos/OpenGL support folder. This is implemented in the DemoApplication class. Each demo derives a class from DemoApplication and implements its own initialization of the physics in the 'initPhysics' method.

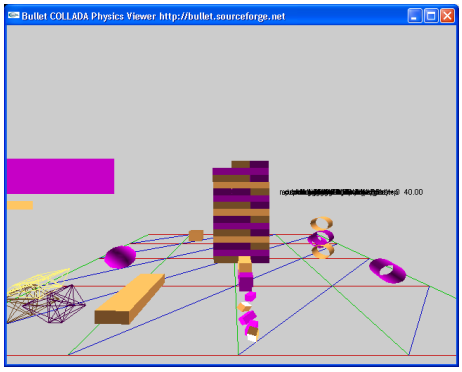
CCD Physics Demo

This is the main demo that shows how to setup a physics simulation, add some objects, and step the simulation. It shows stable stacking, and allows mouse picking and shooting boxes to collapse the wall. The shooting speed of the box can be changed, and for high velocities, the CCD feature can be enabled to avoid missing collisions.



COLLADA Physics Viewer Demo

Imports and exports COLLADA Physics files. It uses the included libxml and COLLADA-DOM library.
The COLLADA-DOM imports a .dae xml file that is generated by tools and plugins for popular 3D modelers. ColladaMaya with Nima from FeelingSoftware, Blender, Ageia's free CreateDynamics tool and other software can export/import this standard physics file format. The ColladaConverter class can be used as example for other COLLADA physics integrations.

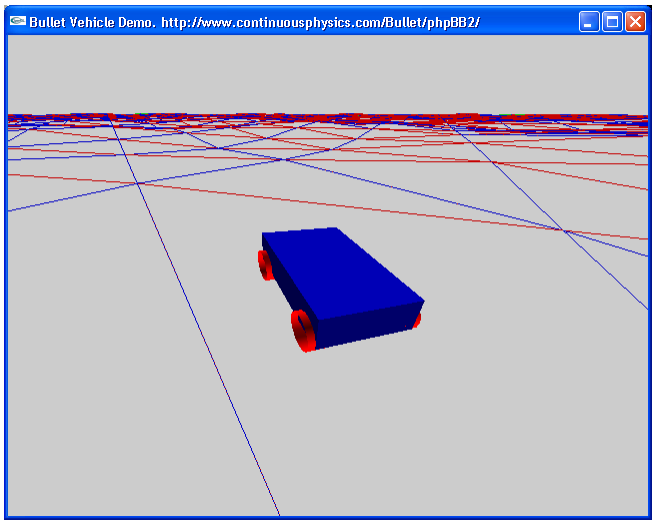


BSP Demo

Import a Quake .bsp files and convert the brushes into convex objects. This performs better then using triangles.

Vehicle Demo

This demo shows the use of the build-in vehicle. The wheels are approximated by ray casts. This approximation works very well for fast moving vehicles. For slow vehicles where the interaction between wheels and environment needs to be more precise the Forklift Demo is more recommended.



Fork Lift Demo

A demo that shows how to use constraints like hinge and generic D6 constraint to build a fork lift vehicle. Wheels are approximated by cylinders.

Low Level Technical Demos

Collision Interfacing Demo

This demo shows how to use Bullet collision detection without the dynamics. It uses the CollisionWorld class, and fills this will CollisionObjects. performDiscreteCollisionDetection method is called and the demo shows how to gather the contact points.

Collision Demo

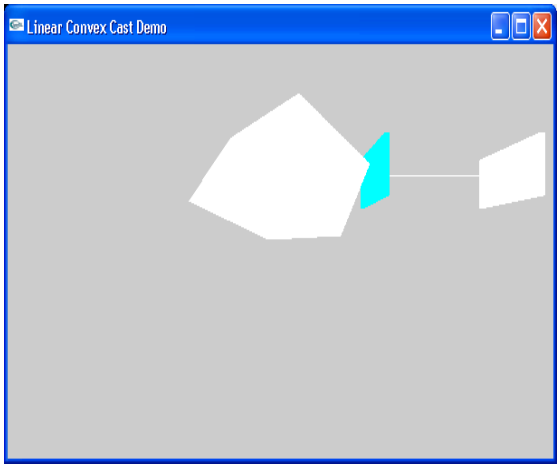
This demo is more low level then previous Collision Interfacing Demo. It directly uses the GJKPairDetector to query the closest points between two objects.

User Collision Algorithm

Shows how you can register your own collision detection algorithm that handles the collision detection for a certain pair of collision types. A simple sphere-sphere case overides the default GJK detection.

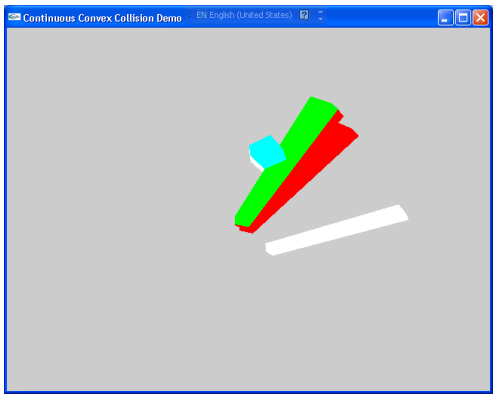
Gjk Convex Cast / Sweep Demo

This demo show how to performs a linear sweep between to collision objects and returns the time of impact. This can be useful to avoid penetrations in camera and character control.



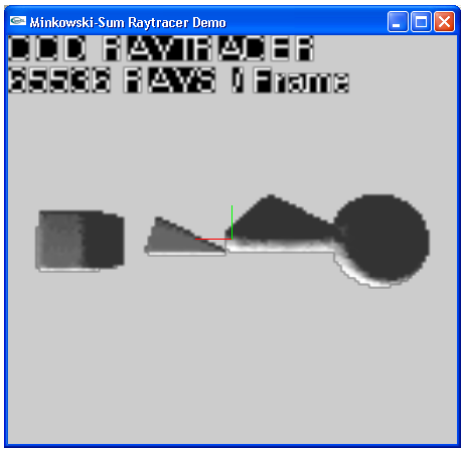
Continuous Convex Collision

Shows time of impact query using continuous collision detection, between two rotating and translating objects. It uses Bullet's implementation of Conservative Advancement.



Raytracer Demo

This shows the use of CCD ray casting on collision shapes. It implements a ray tracer that can accurately visualize the real representation of collision shapes. This includes the collision margin, convex hulls of implicit objects, minkowski sums and other shapes that are hard to visualize.



Concave Demo

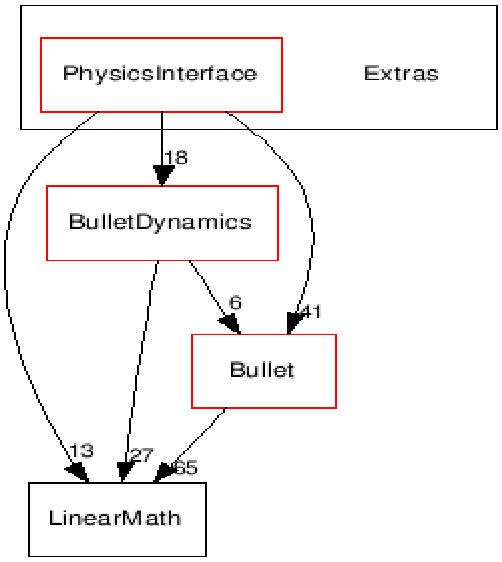
This advanced demo shows how to implement user defined per-triangle restitution and friction in a static triangle mesh. A callback can be registered and triangle identifiers can be used to modify the friction in each reported contact point.

Simplex Demo

This is a very low level demo testing the inner workings of the GJK sub distance algorithm. This calculates the distance between a simplex and the origin, which is drawn with a red line. A simplex contains 1 up to 4 points, the demo shows the 4 point case, a tegrahedron. The Voronoi simplex solver is used, as described by Christer Ericson in his collision detection book.

Bullet Collision Detection and Physics Architecture

Bullet provides Collision Detection and Rigid Body dynamics. The C++ software is divided into several modules with clean dependencies:



The division of those modules is reflected in Bullet's directory structure, and further subdirectories are provided per module. This means that the Collision Detection module can be used without using the BulletDynamics module.

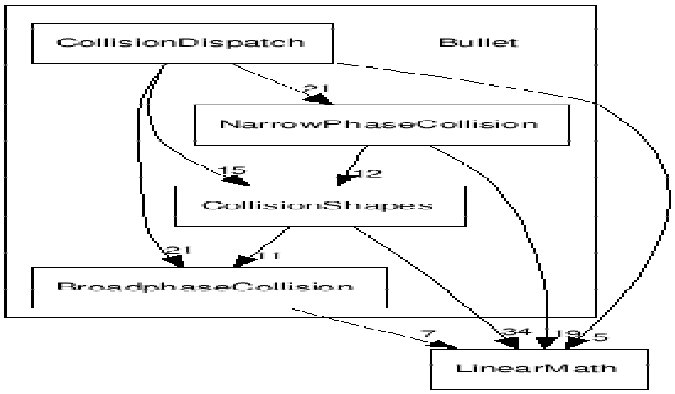
Different implementations of the btDynamicsWorld are available. Most basic version is btSimpleDynamicsWorld. Another is the btDiscreteDynamicsWorld, which tests collision detection at discrete intervals. A btContinuousDynamicsWorld and btParallelDynamicsWorld is under development, and so is a btGpuDynamicsWorld, which benefits from GPU hardware acceleration.

Bullet Collision Detection

The main queries provided by the Collision Detection:

- ✓ Closest Distance and closest points
- ✓ Penetration depth calculation
- ✓ Ray cast
- ✓ Sweep API for casting shapes to find Time of Impact (TOI) along a linear path
- ✓ Time of Impact for Continuous Collision Detection including rotations

Supported Collision Shapes include Box, Sphere, Cylinder, Capsule, Minkowski Sum, Convex Hull, (Concave) Triangle Mesh and Compound Shapes and more.



Additional functionality are related to performance and to provide more detail and information useful for the usage in rigid body dynamics and for AI queries in games. The collision pipeline includes 3 stages: Broadphase, Midphase and Narrowphase.

- ✓ Broadphase

Broadphase provides all overlapping pairs based on axis aligned bounding box (AABB). It includes an efficient culling of all potential pairs using the incremental sweep and prune algorithm.

- ✓ Midphase

The midphase performs additional culling for complex collision shapes like compound shapes and static concave triangle meshes. Bullet uses an optimized Bounding Volume Hierarchy, based on a AABB tree and stackless tree traversal. This traversal provides primitives that need to be tested by the Narrowphase.

- ✓ Narrowphase

The Narrowphase perform the actual distance, penetration or time of impact query. Contact points are collected and maintained over several frames in a persistent way. This means that additional information useful for rigid body simulation can be stored in each contact point. Also this means that algorithms that only provide one contact point at a time can still be used, by gathering additional contact points and performing contact point reduction.

Further documentation and references

Bullet Physics website provides most information:
Visit http://bullet.sf.net or http://continuousphysics.com

On this website there is online doxygen documentation, a wiki with frequently asked questions and tips, and most important a discussion forum.

A paper describing the Bullet's Continuous Collision Detection method is available online at http://continuousphysics.com/BulletContinuousCollisionDetection.pdf

For physics tools and COLLADA physics visit http://www.khronos.org/collada
You can find the latest plugin versions and other information at the COLLADA forum at https://collada.org/public_forum/

Links

COLLADA-DOM included in Bullet: http://colladamaya.sourceforge.net

ColladaMaya plugin http://www.feelingsoftware.com

Blender 3D modeler includes Bullet and COLLADA physics: http://www.blender.org

Ageia CreateDynamics tool http://www.amillionpixels.us/CreateDynamics.zip

This great tool can perform automatic convex decomposition and create ragdolls from graphics skeletons. It is also available from Ageia support forums at http://devsupport.ageia.com

Books

Realtime Collision Detection, Christer Ericson
http://www.realtimecollisiondetection.net/
Bullet uses the discussed voronoi simplex solver for GJK

Collision Detection in Interactive 3D Environments, Gino van den Bergen
http://www.dtecta.com also website for Solid collision detection library
Discusses GJK and other algorithms, very useful to understand Bullet

Physics Based Animation, Kenny Erleben
http://www.diku.dk/~kenny/
Very useful to understand Bullet Dynamics and constraints